(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号

特開平10－49370

(43)公開日　平成10年(1998)２月20日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ　9/38	３１０		Ｇ０６Ｆ　9/38	３１０Ｆ	

審査請求　未請求　請求項の数６　ＯＬ　（全 20 頁）

(21)出願番号　特願平8－203675

(22)出願日　平成８年(1996)８月１日

(71)出願人　000006013
三菱電機株式会社
東京都千代田区丸の内二丁目２番３号

(72)発明者　エドガー・ホルマン
東京都千代田区丸の内二丁目２番３号　三菱電機株式会社内

(72)発明者　吉田　豊彦
東京都千代田区丸の内二丁目２番３号　三菱電機株式会社内

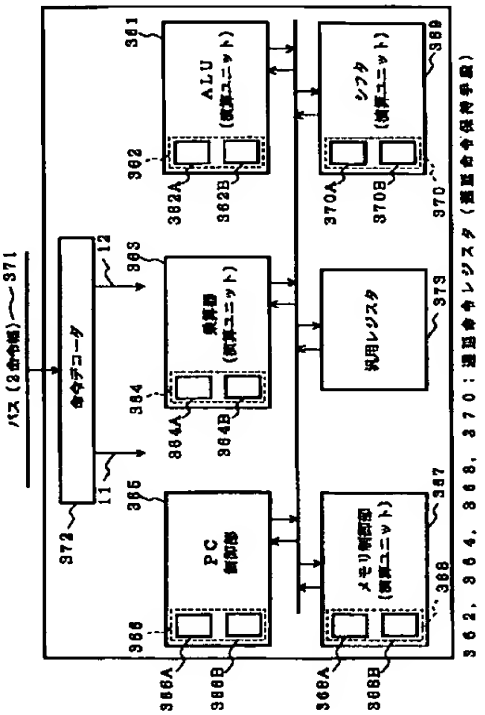
(74)代理人　弁理士　田澤　博昭　（外２名）

(54)【発明の名称】　遅延命令を有するマイクロプロセッサ

(57)【要約】

【課題】　遅延命令は分岐命令に限られているので、効果的な命令のスケジューリングができないという課題があった。

【解決手段】　ＡＬＵ３６１、乗算器３６３、ＰＣ制御部３６５、メモリ制御部３６７およびシフタ３６９は、遅延命令のデコード結果をレジスタ３６２Ａ，３６４Ａ，３６６Ａ，３６８Ａ，３７０Ａに格納するとともに、遅延命令で指定された遅延量に応じたプログラムカウンタ値をレジスタ３６２Ｂ，３６４Ｂ，３６６Ｂ，３６８Ｂ，３７０Ｂに格納するものである。



1

【特許請求の範囲】

【請求項1】　命令をデコードする命令デコーダと、前記命令デコーダの出力に従って命令を実行する命令実行部とを備えた遅延命令を有するマイクロプロセッサにおいて、前記命令実行部におけるプログラムカウンタ値を制御するPC制御部は、遅延分岐命令の分岐先を示す値および遅延分岐命令で指定された遅延量に応じたプログラムカウンタ値を保持する遅延分岐命令保持手段を備えたことを特徴とする遅延命令を有するマイクロプロセッサ。

【請求項2】　命令をデコードする命令デコーダと、前記命令デコーダの出力に従って命令を実行する命令実行部とを備えた遅延命令を有するマイクロプロセッサにおいて、前記命令実行部における演算ユニットは、遅延演算命令による演算内容および遅延演算命令による演算の実行開始を示す値を保持する遅延命令保持手段を備えたことを特徴とする遅延命令を有するマイクロプロセッサ。

【請求項3】　遅延命令保持手段は、遅延演算命令で指定された遅延量に応じた値を保持することを特徴とする請求項2記載の遅延命令を有するマイクロプロセッサ。

【請求項4】　遅延命令保持手段は、遅延演算命令で指定された遅延量に応じたプログラムカウンタ値を保持することを特徴とする請求項3記載の遅延命令を有するマイクロプロセッサ。

【請求項5】　各演算ユニットは、複数の遅延命令保持手段を有する請求項3または請求項4記載の遅延命令を有するマイクロプロセッサ。

【請求項6】　命令実行部は、複数演算を同時に実行する請求項1から請求項5のうちのいずれか1項記載の遅延命令を有するマイクロプロセッサ。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】この発明は、実行時期を遅らせて実行される遅延命令を有するマイクロプロセッサに関するものである。

【0002】

【従来の技術】図20は従来のパイプライン制御方式によるマイクロプロセッサの処理シーケンスを示すシーケンス図である。図において、300は分岐命令、301はパイプラインにおける命令フェッチステージ、302はパイプラインにおけるデコードステージ、303はパイプラインにおける命令実行ステージ、304はパイプラインにおけるライトバックステージ、305は第1の遅延スロットにおける命令、306は第2の遅延スロットにおける命令、307は分岐先で実行される命令を示す。なお、以下、単に「分岐命令」と表現された場合には、それは、プログラムカウンタ（PC）値にオペランドで示されるオフセット値が加算された値であるアドレスに分岐する命令と、オペランドで示されるアドレスに間接的にまたは直接に分岐する命令とを含む。分岐命令とジャンプ命令とが、併記された場合には、それらは、それぞれ、プログラムカウンタ値にオペランドで示されるオフセット値が加算された値であるアドレスに分岐する分岐命令、オペランドで示されるアドレスに間接的にまたは直接に分岐するジャンプ命令を意味する。また、以下、分岐命令およびジャンプ命令は、サブルーチンコール命令を含むものとして説明を進める。

2

【0003】次に動作について説明する。分岐命令が実行されると、図20に示されたパイプライン制御を行うマイクロプロセッサは、分岐先アドレスを3番目のステージである命令実行ステージ303で得ることができる。その時点では、第1の遅延スロットにおける命令305および第2の遅延スロットにおける命令306は、既にデコードステージ302または命令フェッチステージ301にある。マイクロプロセッサは、それらの命令を無効なものとして扱わなくてはならず、パイプラインの無駄が生ずる。

【0004】パイプラインの無駄を防止するために、「コンピュータアーキテクチャ：量的アプローチ（Computer Architecture：A Quantitative Approach），Morgan Kaufmann社，1990年」等の文献で種々の手法が提案されている。例えば、命令のスケジューリング、または命令のスケジューリングと遅延分岐命令（ディレイド分岐命令）との組み合わせによって、パイプラインの無駄を低減することができる。

【0005】例えば、特開平6－131180号公報や特開平6－274352号公報に、ディレイド分岐命令に関する技術が記載されている。一般にディレイド分岐命令における遅延量はマイクロプロセッサのアーキテクチャに応じた固定的な値であるが、特開平6－131180号公報には任意の遅延スロット数を指定できる命令が開示されている。指定された遅延スロット数はデクリメントカウンタに設定される。デクリメントカウンタの値は動作クロック信号の入力に従って減り、値が「1」になると分岐先命令のフェッチが開始される。

【0006】図21は2演算を同時に行うマイクロプロセッサにおける命令デコーダおよび命令実行部の部分の一般的な構成を示すブロック図である。図において、341は算術論理演算を実行するALU、342は乗算を実行する乗算器、343はPC値を計算するPC制御部、344はアドレス計算を行うメモリ制御部、345はシフト演算を実行するシフタ、346は1サイクルで2命令を転送できるバス、347は命令をデコードして命令実行部に制御信号11，12を与えるデコーダ、348は汎用レジスタである。

【0007】図22はプログラムの一例を示す説明図である。図において、ADD，SRA，SUB，MULおよびJMPは、それぞれ、加算命令、シフト命令、減算

命令、乗算命令およびジャンプ命令を示す。加算命令、シフト命令、減算命令、乗算命令およびジャンプ命令は、それぞれ、命令実行部におけるＡＬＵ３４１、シフタ３４５、ＡＬＵ３４１、乗算器３４２およびＰＣ制御部３４３において扱われる。また、例えば、（ｒ３，ｒ０，６）は、汎用レジスタ３４８中のｒ０レジスタの値と即値「６」とを対象とした演算結果を汎用レジスタ３４８中のｒ３レジスタに設定することを示す。

【０００８】図２０に示されたマイクロプロセッサは２演算命令を扱えるので、命令実行部におけるリソースが競合しない限り、図２２に示されたプログラムを２演算命令による各命令に変換したものを扱える。例えば、図２３に示されるように変換された各命令を実行できる。図２３において、各行は２演算命令の１命令に対応する。すなわち、各行は同時実行される２演算を示す。ＳＲＡとＳＵＢとは、リソースの競合はないがレジスタ依存の関係にあるので、同時に実行することはできない。よって、2行目にＮＯＰが置かれている。このような命令のスケジューリングは、プログラマまたはコンパイラによってなされる。

【０００９】

【発明が解決しようとする課題】従来の遅延命令を有するマイクロプロセッサは以上のように構成されているので、以下のような課題があった。

（１）遅延量が固定的にしか指定できないディレイド分岐命令を有するマイクロプロセッサでは、効果的な命令のスケジューリングができない。例えば、図２３における５行目のＪＭＰを削除して２行目のＮＯＰの位置に遅延量「２」のディレイドジャンプ命令を置くことは効果的である。そのようにすれば、図２３の４行目の命令のフェッチ後に直ちにジャンプ先アドレスＴＧＴの命令のフェッチが開始され、パイプラインの無駄が生じないからである。遅延量「２」しか指定できないマイクロプロセッサでは、そのようなディレイド分岐命令を置いた命令群を実行できる。しかし、例えば遅延量「３」しか指定できないマイクロプロセッサを使用する場合には、プログラマまたはコンパイラは、そのようなスケジューリングを行うことはできない。

（２）指定された遅延量に応じた値をデクリメントカウンタに設定するマイクロプロセッサでは、命令遅延中に割り込みや新たな分岐が生じた場合に矛盾なく処理を進めるための構成が複雑になる。例えば、デクリメントカウンタの値は動作クロックに従ってカウントダウンされるので、何等の考慮も払わないと、設定された遅延量と実際の遅延量との間で、割り込み処理等で費やされた動作クロック数分の狂いが生ずる。

（３）ディレイド命令は分岐命令に限られているので、効果的な命令のスケジューリングができない。

【００１０】この発明は上記のような課題を解決するためになされたもので、処理の矛盾を生じさせることなく任意の遅延量を指定できるディレイド命令を扱えて効果的な命令のスケジューリングができる環境をプログラマに提供でき、その結果、プログラムをより高速に実行しうるマイクロプロセッサを得ることを目的とする。

【００１１】

【課題を解決するための手段】請求項１記載の発明に係る遅延命令を有するマイクロプロセッサは、ＰＣ値を制御するＰＣ制御部が、ディレイド分岐命令の分岐先アドレスを示す値およびディレイド分岐命令で指定された遅延量に応じたプログラムカウンタ値を保持する遅延分岐命令保持手段を有するものである。

【００１２】請求項２記載の発明に係る遅延命令を有するマイクロプロセッサは、各演算ユニットが、ディレイド演算命令の内容およびディレイド演算命令の固定のまたは可変の遅延量に応じた値を保持する遅延命令保持手段を有するものである。ここで、ディレイド演算命令の内容とは、ディレイド演算命令のデコード結果であって、いかなる演算を行うかを示す情報である。なお、特に断らない限り、ディレイド分岐命令は、ディレイド演算命令に含まれないものとする。

【００１３】請求項３記載の発明に係る遅延命令を有するマイクロプロセッサは、任意の遅延量を指定できるディレイド演算命令で指定された遅延量に応じた値を保持する遅延命令保持手段を有するものである。

【００１４】請求項４記載の発明に係る遅延命令を有するマイクロプロセッサは、各演算ユニットが、ディレイド演算命令で指定された遅延量に応じたＰＣ値を保持する遅延命令保持手段を有するものである。

【００１５】請求項５記載の発明に係る遅延命令を有するマイクロプロセッサは、各演算ユニットに、複数の遅延命令保持手段が設けられているものである。

【００１６】請求項６記載の発明に係る遅延命令を有するマイクロプロセッサは、上記の各構成を有するとともに、命令実行部が複数演算を同時に実行するものである。

【００１７】

【発明の実施の形態】以下、この発明の実施の一形態を説明する。

実施の形態１．図１はこの発明の実施の形態１によるマイクロプロセッサの構成を示すブロック図である。このマイクロプロセッサは、３２ビットの内部データバスを有する３２ビットマイクロプロセッサである。図において、２は命令ＲＡＭ６から６４ビット幅のＩＤバスを介して入力した命令コードをデコードする処理を行う命令デコードユニット（命令デコーダ）、３はアドレス計算を行うメモリユニット（命令実行部）、４は論理演算やシフト演算を行う整数演算ユニット（命令実行部）、５は３２ビット×６４ワードの汎用レジスタ、７はデータが格納されるデータＲＡＭである。

【００１８】命令デコードユニット２において、８，９

はそれぞれ命令コードをデコードするデコーダ、１０はプロセッサの状態を示すプロセッサ状態語（Processor Status Word 、以下、プロセッサ状態語をＰＳＷと呼ぶ）である。命令デコードユニット２は、さらに、デコーダ８のデコード結果とＰＳＷ１０の内容にもとづいて制御信号１１を作成し、それをメモリユニット３に与える。また、命令デコードユニット２は、デコーダ９のデコード結果とＰＳＷ１０の内容にもとづいて制御信号１２を作成し、それを整数演算ユニット４に与える。

【００１９】メモリユニット３において、１３はジャンプや分岐を含まない命令を実行するとＰＣ値に８を加えて次に実行する命令に対するＰＣ値を算出するとともに、ジャンプや分岐を含む命令の実行時に分岐変位をＰＣ値に加算したり、演算で指定されたアドレッシングモードに応じた計算を行ってジャンプ先の命令に対するＰＣ値を計算するＰＣ制御部である。また、ＰＣ制御部１３は、計算したＰＣ値を３２ビット幅のＩＡバスを介して命令ＲＡＭ６に送り、命令ＲＡＭ６から命令コードを出力させる。１４はオペランドとなるデータのアクセスを制御するメモリ制御部である。メモリ制御部１４は、３２ビット幅のＤＡバスを介してアドレスデータをデータＲＡＭ７に転送し命令実行に必要なデータを６４ビット幅のＤＤバスを介してアクセスする。１５は汎用レジスタ５から３２ビット幅のＳ１バス、Ｓ２バス、Ｓ３バスを介して転送された最大３ワードのデータを用いて算術論理演算を行い演算結果を３２ビット幅のＤ１バスを介して汎用レジスタ５に転送するＡＬＵ、１６は汎用レジスタ５からＳ１バス、Ｓ２バス、Ｓ３バスを介して転送されたデータを用いてシフト演算を行い演算結果をＤ１バスを介して汎用レジスタ５に転送するシフタである。

【００２０】メモリユニット３に対して、Ｓ１バス、Ｓ２バス、Ｓ３バス、Ｓ４バスを介して、３２ビット長のデータを一時に４ワード転送することが可能である。従って、例えば、第１のレジスタの内容と第２のレジスタの内容との和でアドレッシングされるメモリの領域に第３のレジスタの内容をストアするとともに、第３のレジスタの内容をストアしたアドレスに所定値を加算して得られる値でアドレッシングされるメモリの領域に第４のレジスタの内容をストアする２ワードストア命令を実現することができる。また、メモリユニット３は、Ｄ１バスおよびＤ２バスを介して、メモリユニット３内での２ワードの演算結果またはデータＲＡＭ７から転送された＊２ワードのデータを汎用レジスタ５に転送することができる。

【００２１】整数演算ユニット４において、１７は汎用レジスタ５から３２ビット幅のＳ４バス、Ｓ５バス、Ｓ６バスを介して転送された最大３ワードのデータを用いて乗算を行い演算結果を３２ビット幅のＤ２バス、Ｄ３バスを介して汎用レジスタ５に転送する乗算器、１８は乗算の結果を累積加算または累積減算して保持するアキュムレータである。アキュムレータとして、６４ビットのものが２本ある。１９は汎用レジスタ５からＳ４バス、Ｓ５バス、Ｓ６バスを介して転送された最大３ワードのデータを用いて算術論理演算を行い演算結果をＤ２バス、Ｄ３バスを介して汎用レジスタ５に転送するＡＬＵ、２０は汎用レジスタ５からＳ４バス、Ｓ５バス、Ｓ６バスを介して転送されたデータを用いてシフト演算を行い演算結果をＤ２バス、Ｄ３バスを介して汎用レジスタ５に転送するシフタである。

【００２２】このマイクロプロセッサでは、汎用レジスタ５から、最大６種類のレジスタ値を読み出すことが可能であって、読み出されたデータは、それぞれ、Ｓ１バス、Ｓ２バス、Ｓ３バス、Ｓ４バス、Ｓ５バス、Ｓ６バスに出力される。また、汎用レジスタ５には、Ｄ１バス、Ｄ２バス、Ｄ３バスを介して最大３種類のレジスタ値を同時に書き込むことが可能である。

【００２３】図２はこのマイクロプロセッサの命令フォーマットを示す説明図である。命令フォーマットとして、図２（ａ）に示すような１つの命令コードで２つの演算（operation ）を指示する２演算命令のフォーマット１０１と、図２（ｂ）に示すような１つの命令コードで１つの演算を指示する１演算命令のフォーマット１０２とがある。２演算命令のフォーマット１０１には、フィールド１０３およびフィールド１０４からなるフォーマットフィールドと、２つの演算フィールド１０６，１０７と、各演算フィールド１０６，１０７に付属する各実行条件フィールド１０５とがある。１演算命令のフォーマット１０２には、フィールド１０３およびフィールド１０４からなるフォーマットフィールドと、演算フィールドと、演算フィールドに付属する実行条件フィールド１０５とがある。演算フィールドは、フィールド１０８，１０９，１１０からなる。

【００２４】フォーマットフィールドは、以下のような意味を示す。

コード：フォーマット	実行の順番 operation_0	operation_1
ＦＭ＝００：２命令	第１	第１
０１：２命令	第１	第２
１０：２命令	第２	第１
１１：１命令	第１	……

ここで、ＦＭは、フィールド１０３およびフィールド１※50※０４からなる２ビットの値である。

【００２５】ＦＭ＝００の場合、この命令は２演算命令であることを示す。そして、演算フィールド１０６で指示されたoperation_０の演算と演算フィールド１０７で指示されたoperation_１の演算とが、デコード直後のクロックサイクルで並列に実行される。operation_０の演算はメモリユニット３で実行され、operation_１の演算は整数演算ユニット４で実行される。ＦＭ＝０１の場合、この命令は２演算命令であることを示す。そして、operation_０の演算が、デコード直後のクロックサイクルで実行され、operation_１の演算が、operation_０の演算に対して、１クロックサイクル遅れて実行される。ＦＭ＝１０の場合、この命令は２演算命令であることを示す。そして、operation_１の演算が、デコード直後のクロックサイクルで実行され、operation_０の演算が、operation_１の演算に対して、１クロックサイクル遅れて実行される。ＦＭ＝１１の場合、この命令は１演算命令であることを示す。そして、フィールド１０８，１０９，１１０からなる演算フィールドで指示された１つの演算がデコード直後のクロックサイクルで実行される。

【００２６】実行条件フィールド１０５は、以下のような意味を持つ。

コード：	実行条件		
ＣＣ＝０００：	常時		
００１：	Ｆ０＝真	かつ	Ｆ１＝無視
０１０：	Ｆ０＝偽	かつ	Ｆ１＝無視
０１１：	Ｆ０＝無視	かつ	Ｆ１＝真
１００：	Ｆ０＝無視	かつ	Ｆ１＝偽
１０１：	Ｆ０＝真	かつ	Ｆ１＝真
１１０：	Ｆ０＝真	かつ	Ｆ１＝偽
１１１：	予約済		

【００２７】実行条件フィールド１０５は、マイクロプロセッサの実行コントロールフラグＦ０，Ｆ１の値に応じて、演算フィールド１０６，１０７のoperation_０の演算やoperation_１の演算、およびフィールド１０８，１０９，１１０からなる演算フィールドの演算が有効であるか無効であるか定める。実行コントロールフラグＦ０，Ｆ１については後で説明する。演算が有効であるとは、演算結果がレジスタ、メモリおよびフラグに反映され、その演算による動作の結果が残ることを意味する。演算が無効であるとは、演算結果がレジスタ、メモリおよびフラグに反映されず、あたかも無効演算（ＮＯＰ）が実行されたかのような動作の結果が残ることを意味する。

【００２８】実行条件フィールド１０５の値ＣＣ＝０００のときには、実行コントロールフラグＦ０，Ｆ１の値にかかわらず常に演算は有効である。ＣＣ＝００１のときには、実行コントロールフラグＦ０＝真のときにのみ演算は有効である。実行コントロールフラグＦ１の状態はいずれでもよい。ＣＣ＝０１０のときには、実行コントロールフラグＦ０＝偽のときにのみ演算は有効である。実行コントロールフラグＦ１の状態はいずれでもよい。ＣＣ＝０１１のときには、実行コントロールフラグＦ１＝真のときにのみ演算は有効である。実行コントロールフラグＦ０の状態はいずれでもよい。ＣＣ＝１００のときには、実行コントロールフラグＦ１＝偽のときにのみ演算は有効である。実行コントロールフラグＦ０の状態はいずれでもよい。ＣＣ＝１０１のときには、実行コントロールフラグＦ０＝真かつＦ１＝真のときにのみ演算は有効である。ＣＣ＝１１０のときには、実行コントロールフラグＦ０＝真かつＦ１＝偽のときにのみ演算は有効である。ＣＣ＝１１１のときの動作は未定義であり、ユーザは、ＣＣ＝１１１となる命令を用いることはできない。

【００２９】図３は演算フィールドの詳細な内容を示す説明図である。フォーマット１１１～１１７は、それぞれ２８ビットで表現される短型の演算フィールド１０６または演算フィールド１０７によるものである。フォーマット１１８は、フィールド１０８，１０９，１１０で構成される長型の演算フィールドによるものである。

【００３０】フォーマット１１１（Ｓｈｏｒｔ＿Ｍ）は、演算内容を指定するフィールド１２０、レジスタ番号を指定する２つのフィールド１２１，１２２、レジスタ番号または６ビット長の即値を指定するフィールド１２３、およびフィールド１２３がレジスタ番号を示すのか即値を示すのかを指定するフィールド１２４で構成される。図３に示すように、フィールド１２４の値Ｘが「００」、「０１」または「１１」であるときにはフィールド１２３がレジスタ番号を示していることを示し、「１０」であるときには即値を示していることを示す。このフォーマット１１１は、レジスタ間接アドレッシングのメモリアクセス演算に用いられる。

【００３１】フォーマット１１２（Ｓｈｏｒｔ＿Ａ）は、演算内容を指定するフィールド１２０、レジスタ番号を指定する２つのフィールド１２１，１２２、レジスタ番号または６ビット長の即値を指定するフィールド１２３、およびフィールド１２３がレジスタ番号を示すのか即値を示すのかを指定するフィールド１２５で構成される。図３に示すように、フィールド１２５の値Ｘ’が「０」であるときにはフィールド１２３がレジスタ番号を示していることを示し、「１」であるときには即値を示していることを示す。このフォーマット１１２は、算術演算、論理演算、シフト演算およびビット演算に用いられる。

【００３２】フォーマット１１３（Ｓｈｏｒｔ＿Ｂ１）は、演算内容を指定するフィールド１２０およびレジスタ番号を指定するフィールド１２６で構成される。このフォーマット１１３は、レジスタ指定によるジャンプ命令および分岐命令に用いられる。フォーマット１１４（Ｓｈｏｒｔ＿Ｂ２）は、演算内容を指定するフィールド１２０および１８ビット長のディスプレイスメントの

フィールド１２７で構成される。このフォーマット１１４は、ジャンプ命令および分岐命令に用いられる。

【００３３】フォーマット１１５（Ｓｈｏｒｔ＿Ｂ３）は、演算内容を指定するフィールド１２０、レジスタ番号を指定するフィールド１２１、レジスタ番号または１２ビット長の即値を指定するフィールド１２８、フィールド１２８がレジスタ番号を示すのか即値を示すのかを指定するフィールド１２９、およびゼロ判定にもとづいてフィールド１２１にもとづく条件ジャンプまたは条件分岐を行うか否か指定するフィールド１３０で構成される。このフォーマット１１５は、条件ジャンプ命令および条件分岐命令に使用される。

【００３４】フォーマット１１６（Ｓｈｏｒｔ＿Ｄ１）は、演算内容を指定するフィールド１２０、レジスタ番号を指定するフィールド１２１、レジスタ番号または１２ビット長の即値を指定するフィールド１２８、フィールド１２８がレジスタ番号を示すのか即値を示すのかを指定するフィールド１２９で構成される。このフォーマット１１６は、条件ジャンプ命令、条件分岐命令およびリピート命令に使用される。フォーマット１１７（Ｓｈｏｒｔ＿Ｄ２）は、演算内容を指定するフィールド１２０、レジスタ番号または１２ビット長の即値を指定するフィールド１２８、フィールド１２８がレジスタ番号を示すのか即値を示すのかを指定するフィールド１２９、遅延命令（ディレイド命令）に関するフィールド１３１で構成される。このフォーマット１１７は、ディレイドジャンプ命令、ディレイド分岐命令およびリピート命令に使用される。

【００３５】フォーマット１１８（Ｌｏｎｇ）は、演算内容を指定するフィールド１２０、レジスタ番号を指定する２つのフィールド１２１，１２２、３２ビット長の即値を指定するフィールド１３２で構成される。このフォーマット１１８は、複雑な算術演算、大きな即値を用いる算術演算、大きなディスプレイスメント付きレジスタ間接アドレッシングのメモリアクセス演算、大きな変位の分岐演算および絶対番地へのジャンプ命令などに使用される。

【００３６】図４はマイクロプロセッサのレジスタ構成を示す説明図である。このマイクロプロセッサは、図４（ａ）に示すような６４本の３２ビット長の汎用レジスタ５、図４（ｂ）に示すような１２本の制御レジスタ１５０、および図４（ｃ）に示すような２本のアキュムレータ１８を持つ。Ｒ０の汎用レジスタ１４０の内容は常に０であり、そこへの書き込みは無視される。Ｒ６２の汎用レジスタは、サブルーチンからの戻り先アドレスが設定されるリンクレジスタである。Ｒ６３の汎用レジスタは、スタックポインタであり、ＰＳＷ１０のＳＭフィールドの値に応じてユーザスタックポインタ（ＳＰＵ）または割り込みスタックポインタ（ＳＰＩ）として動作＊する。制御レジスタ１５０には、プログラムカウンタ１５１、ＰＳＷ１０、および各種の専用レジスタが含まれる。図３に示すフォーマット１１２による演算では、６４本の汎用レジスタ５のそれぞれを上位１６ビットと下位１６ビットとに分けてアクセスできる。また、２本のアキュムレータ１８を、上位３２ビットと下位３２ビットとに分けて別々にアクセスできる。

【００３７】図５はＰＳＷ１０の詳細内容を示す説明図である。ＰＳＷ１０の上位１６ビットには、スタックポインタを切り替えるためのＳＭフィールド１７１、セルフデバッグトラップ（ＳＤＢＴ）の検出を示すＥＡフィールド１７２、ＳＤＢＴの許可を指定するＤＢフィールド１７３、割り込み許可を指定するＩＥフィールド１７４、リピート動作の許可を指定するＲＰフィールド１７５、モジュロアドレッシングの許可を指定するＭＤフィールド１７６がある。下位１６ビットはフラグフィールド１８０である。フラグフィールド１８０には８個のフラグがあり、その中のＦ０フラグ１８１およびＦ１フラグ１８２は演算の有効／無効を指定する。各フラグの値は比較演算や算術演算の結果に応じて変化する。また、フラグ初期化演算で初期化したり、フラグ値書き込み演算で任意の値をフラグフィールド１８０に書き込むことによって変化する。フラグフィールド１８０の内容は、フラグ値読み出し演算によって読み出される。

【００３８】各フラグは、以下のような意味を有する。

ＳＭ＝０　：スタックモード０→ＳＰＩを使用
ＳＭ＝１　：スタックモード１→ＳＰＵを使用
ＥＡ＝０　：ＳＤＢＴを未検出
ＥＡ＝１　：ＳＤＢＴを検出
ＤＢ＝０　：ＳＤＢＴを非許可
ＤＢ＝１　：ＳＤＢＴを許可
ＩＥ＝０　：割り込み非許可
ＩＥ＝１　：割り込み許可
ＲＰ＝０　：リピートブロック無効
ＲＰ＝１　：リピートブロック有効
ＭＤ＝０　：モジュロアドレッシング無効
ＭＤ＝１　：モジュロアドレッシング有効
Ｆ０　：汎用フラグ（実行コントロールフラグ）
Ｆ１　：汎用フラグ（実行コントロールフラグ）
Ｆ２　：汎用フラグ
Ｆ３　：汎用フラグ
Ｆ４（Ｓ）　：飽和演算フラグ
Ｆ５（Ｖ）　：オーバーフローフラグ
Ｆ６（ＶＡ）：累積オーバーフローフラグ
Ｆ７（Ｃ）　：キャリー／ボローフラグ

【００３９】以下、このマイクロプロセッサの命令一覧を示す。

Ａ．マイクロプロセッサ機能に関する命令

Ａ－１．ロード／ストア命令

11

LDB　　:Load one byte to a register with sign extension
[1バイトロード（符号拡張あり）]

LDBU　　:Load one byte to a register with zero extension
[1バイトロード（ゼロ拡張あり）]

LDH　　:Load one half-word to a register with sign extension
[1ハーフワードロード（符号拡張あり）]

LDHH　　:Load one half-word to a register high
[1ハーフワードロード（レジスタ上位へ）]

LDHU　　:Load one half-word to a register with zero extension
[1ハーフワードロード（ゼロ拡張あり）]

LDW　　:Load one word to a register
[1ワードロード]

LD2W　　:Load two words to registers
[2ワードロード]

LD4BH　　:Load four bytes to four half-words in two registers with sign extension
[4バイトロード（2レジスタへ，符号拡張あり）]

LD4BHU　　:Load four bytes to four half-words in two registers with zero extension
[4バイトロード（2レジスタへ，ゼロ拡張あり）]

LD2H　　:Load two half-words to two words in two registers with sign extension
[2ハーフワードロード（2レジスタへ，符号拡張あり）]

STB　　:Store one byte from a register
[1バイトストア]

STH　　:Store one half-word from a register
[1ハーフワードストア]

STHH　　:Store one half-word from a register high
[1ハーフワードストア（レジスタ上位から）]

STW　　:Store one word from a register
[1ワードストア]

ST2W　　:Store two words from registers
[2ワードストア]

ST4HB　　:Store four bytes from four half-words from two registrers
[4バイトストア（2レジスタの4ハーフワードから）]

ST2H　　:Store two half-words from two registrers
[2ハーフワードストア（2レジスタから）]

MODDEC　　:Decrement a register value by a 5-bits immediate value
[即値5ビットのデクリメント]

MODINC　　:Increment a register value by a 5-bits immediate value
[即値5ビットのインクリメント]

【0040】

A－2．転送命令

MVFSYS　　:Move a control register to a general purpose register
[制御レジスタから汎用レジスタへ]

MVTSYS　　:Move a general purpose register to a control register
[汎用レジスタから制御レジスタへ]

MVFACC　　:Move a word from an accumulator
[アキュムレータからの1ワード転送]

MVTACC　：Move two general purpose registers to an accumulator
［2汎用レジスタ内容のアキュムレータへの転送］

【0041】

A－3．比較命令

CMPcc　：Compare ［比較］
cc＝EQ（等しい），NE（不等），GT（より大），
GE（以上），LT（未満），LE（以下），
PS（ともに正），NG（ともに負）

CMPUcc　：Compare unsigned［比較（符号なし）］
cc＝GT，GE，LT，LE

【0042】A－4．最大値／最小値命令　　＊【0043】
reserved［予約済］　　＊

A－5．算術演算命令

ABS　：Absolute［絶対値をとる］
ADD　：Add ［加算］
ADDC　：Add with carry［加算（キャリー付き）］
ADDHppp：Add half-word ［ハーフワード加算］
ppp＝LLL（レジスタ下位，レジスタ下位，レジスタ下位），LLH（レジスタ下位，レジスタ下位，レジスタ上位），LHL，LHH，HLL，HLH，HHL，HHH
ADDS　：Add register Rb with the sign of the third operand
［レジスタRbに符号を付ける］
ADDS2H　：Add sign to two half-words
［2ハーフワードに符号を付ける］
ADD2H　：Add two pairs of half-words
［2ハーフワード同士の加算］
AVG　：Average with rounding towards positive infinity
［平均をとる］
AVG2H　：Average two pairs of half-words rounding
towards positive infinity
［2ハーフワードそれぞれの平均をとる］
JOINpp　：Join two half-words ［2ハーフワードの結合］
pp＝LL，LH，HL，HH
SUB　：Subtract［減算］
SUBB　：Subtract with borrow［ボロー付き減算］
SUBHppp：Subtract half-word［ハーフワードの減算］
ppp＝LLL，LLH，LHL，LHH，HLL，HLH，HHL，HHH
SUB2H　：Subtract two pairs of half-words
［2つのハーフワードの減算］

【0044】

A－6．論理演算命令

AND　：logical AND ［論理積］
OR　：logical OR ［論理和］
NOT　：logical NOT ［反転］
XOR　：logical exclusive OR［排他的論理和］
ANDFG　：logical AND flags ［フラグの論理積］
ORFG　：logical OR flags［フラグの論理和］
NOTFG　：logical NOT a flag［フラグの反転］

XORFG　：logical exclusive OR flags［フラグの排他的論理和］

【0045】

A－7．シフト演算命令

SRA　：Shift right arithmetic［算術右シフト］
SRA2H　：Shift right arithmetic two half-words
［2つのハーフワードの算術右シフト］
SRC　：Shift right concatenated registers
［レジスタ連鎖右シフト］
SRL　：Shift right logical ［論理右シフト］
SRL2H　：Shift right logical two half-words
［2つのハーフワードの論理右シフト］
ROT　：Rotate right［右回転］
ROT2H　：Rotate right two half-words
［2つのハーフワードの右回転］

【0046】A－8．ビット操作命令

＊BSET　：Set a bit ［ビットセット］
BCLR　：Clear a bit ［ビットクリア］
BTST　：Test a bit［ビットテスト］
BNOT　：Invert a bit［ビット反転］

＊　【0047】

A－9．分岐命令

BRA　：Branch［分岐］
BRATZR　：Branch if zero［ゼロなら分岐］
BRATNZ　：Branch if not zero［ゼロでないなら分岐］
BSR　：Branch to subroutine［サブルーチンへ分岐］
BSRTZR　：Branch to subroutine if zero
［ゼロならサブルーチンへ分岐］
BSRTNZ　：Branch to subroutine if not zero
［ゼロでないならサブルーチンへ分岐］
JMP　：Jump［無条件ジャンプ］
JMPTZR　：Jump if zero［ゼロならジャンプ］
JMPTNZ　：Jump if not zero［ゼロでないならジャンプ］
JSR　：Jump to subroutine［サブルーチンへジャンプ］
JSRTZR　：Jump to subroutine if zero
［ゼロならサブルーチンへジャンプ］
JSRTNZ　：Jump to subroutine if not zero
［ゼロでないならサブルーチンへジャンプ］
NOP　：No Operation［無操作］

［ディレイド分岐，ジャンプ命令］

DBRA　：Delayed branch［ディレイド分岐］
DBRAI　：Delayed branch immediate［ディレイド分岐（即値）］
DBSR　：Delayed branch to subroutine
［ディレイドサブルーチン分岐］
DBSRI　：Delayed branch immediate to subroutine
［ディレイドサブルーチン分岐（即値）］
DJMP　：Delayed jump［ディレイドジャンプ］
DJMPI　：Delayed jump immediate［ディレイドジャンプ（即値）］
DJSR　：Delayed jump to subroutine
［ディレイドサブルーチンジャンプ］
DJSRI　：Delayed jump immediate to subroutine
［ディレイドサブルーチンジャンプ（即値）］

【0048】

A－10．OS関連命令

TRAP　：Trap［トラップ］

REIT　：Return from exception, interrupts and traps
［例外、割り込み、トラップからのリターン］

【0049】

B．DSP機能に関する命令

B－1．算術操作命令

MUL　：Multiply［乗算］

MULX　：Multiply with extended precision［倍精度乗算］

MULXS　：Multiply and shift to the right by one with extended precision
［倍精度乗算および1ビット右シフト］

MULX2H　：Multiply two pairs of half-words with extended precision
［2ハーフワードずつの倍精度乗算］

MULHXpp：Multiply two half-words with extended precision
pp＝LL, LH, HL, HH
［2ハーフワードの倍精度乗算］

MUL2H　：Multiply two pairs of half-words
［2ハーフワードずつの乗算］

MACa　：Multiply and add［積和演算］
a（アキュムレータ指定）＝0，1

MACSa　：Multiply, shift to the right by one and add
a＝0，1
［1ビット右シフト付き積和演算］

MSUBa　：Multiply and subtract［積和（減算）演算］
a＝0，1

MSUBSa　：Multiply, shift to the right by one and subtract
a＝0，1
［1ビット右シフト付き積和（減算）演算］

【0050】

B－2．リピート命令

REPEAT　：Repeat a block of instructions
［命令ブロックの繰り返し］

REPEATI：Repeat a block of instructions immediate
［命令ブロックの繰り返し（即値指定）］

【0051】図6はマイクロプロセッサの並列2命令実行時のパイプライン動作を示す説明図である。この動作は、命令のフォーマットフィールドの値FM＝00のときに実行される。パイプライン190，195は、命令フェッチステージ191、デコード／アドレス演算ステージ192、実行／メモリアクセスステージ193およびライトバックステージ194で構成される。並列2命令実行時には、メモリユニット3での実行と整数演算ユニット4での実行とが並列に行われる。図7はマイクロプロセッサのシーケンシャル命令実行時のパイプライン動作を示す説明図である。この動作は、命令のフォーマットフィールドの値FM＝01，10，11のときに実行される。パイプライン200は、命令フェッチステージ、デコード／アドレス演算ステージ、実行／メモリアクセスステージ、およびライトバックステージで構成されるが、この場合には、メモリユニット3での実行と整数演算ユニット4での実行とのうちのいずれかが、一時に実行される。なお、図6および図7に示されたパイプライン動作では、2番目のステージであるデコード／アドレス演算ステージ192において分岐先アドレスが得られる。

【0052】図8はこの発明の実施の形態1によるマイクロプロセッサの構成を示すブロック図である。図において、366は分岐命令のデコード値を保持するための

レジスタ１３Ａと分岐命令が実行されるべき時期に対応したＰＣ値を保持するレジスタ１３Ｂとを含む遅延分岐命令レジスタ（遅延分岐命令保持手段）である。分岐命令が実行されるべき時期に対応したＰＣ値とは、マイクロプロセッサのＰＣの値がその値になるとディレイド分岐命令を実行することになる値である。なお、既に定義したように、単に「分岐命令」と表現した場合には、厳密な意味での分岐命令とジャンプ命令との双方の概念を含む。

【００５３】図９はディレイド分岐命令の基本的なフォーマット３２０を示す説明図である。基本的には、ディレイド分岐命令のフォーマット３２０は、オペコード３２１、遅延量を指定するフィールド３２２および分岐先アドレスを指定するためのオフセットまたはアドレスが指定されるフィールド３２３を持つ。ディレイド分岐命令は、例えば、図３に示すフォーマット１１６（Ｓｈｏｒｔ＿Ｄ１）、フォーマット１１７（Ｓｈｏｒｔ＿Ｄ２）またはフォーマット１１８（Ｌｏｎｇ）で実現される。フォーマット１１６（Ｓｈｏｒｔ＿Ｄ１）は、遅延量としてレジスタ設定値が用いられる場合に使用される。フォーマット１１７（Ｓｈｏｒｔ＿Ｄ２）は、遅延量として即値が用いられる場合に使用される。フォーマット１１８（Ｌｏｎｇ）は、分岐先アドレスを３２ビット即値で指定する場合に使用される。各フォーマットにおいて、オペコードはフィールド１２０で指定される。また、フィールド１２９は、フィールド１２８がレジスタ番号を示すのか即値を示すのかを指定するために使用される。フィールド１２１は、ＤＢＲＡ，ＤＢＳＲ，ＤＪＭＰ，ＤＪＳＲの各命令において遅延量がレジスタで指定されるときのレジスタ指定領域として使用され、フィールド１３１は、遅延量を指定する即値の領域として使用される。

【００５４】図１０は幾つかのディレイド分岐命令の例を示す説明図である。命令３２４は、遅延量を即値で指定するとともに分岐のオフセットを即値で指定する命令である。命令３２５は、遅延量をレジスタで指定するとともに分岐のオフセットを即値で指定する命令である。命令３２６は、遅延量を即値で指定するとともに分岐のオフセットをレジスタで指定する命令である。命令３２７は、遅延量をレジスタで指定するとともに分岐のオフセットをレジスタで指定する命令である。

【００５５】図１１はディレイド分岐命令３３０、ディレイドジャンプ命令３３２およびディレイドサブルーチンコール命令３３１，３３３が、同一フォーマットでどのように実現されるのかを示すための説明図である。

【００５６】次に動作について説明する。命令がディレイド分岐命令であったことを命令デコードユニット２のデコーダ８が認識すると、命令デコードユニット２は、そのことを示す制御信号１１をメモリユニット３に出力する。メモリユニット３において、ＰＣ制御部１３は、制御信号１１によって入力した命令デコード結果をレジスタ１３Ａに格納する。従って、レジスタ１３Ａには、ディレイド分岐命令による分岐先を示す情報等が格納される。また、ＰＣ制御部１３は、分岐命令が実行されるべき時期に応じたＰＣ値をレジスタ１３Ｂに格納する。

【００５７】ＰＣ制御部１３は、マイクロプロセッサにおける実際のＰＣ値がレジスタ１３Ｂに格納されている値と一致したことを検知したら、レジスタ１３Ａに格納されている分岐先情報にもとづいて分岐命令を実行する。すなわち、レジスタ１３Ａに格納されている分岐先情報が示す値をＰＣに設定する。この結果、レジスタ１３Ｂに格納されている値に応じた数の命令がフェッチされたら、次のサイクルで、分岐先にある命令がフェッチされる。

【００５８】図１２はディレイド分岐命令を含むプログラムの一例を示す説明図である。図１２における２行目のＤＢＲＡ命令は、遅延量「３」を指定するとともに分岐先としてＴＧＴ１を指定する命令である。命令デコードユニット２において、ＤＢＲＡ命令がデコードされると、ＰＣ制御部１３は、分岐先を示すＴＧＴ１に応じた情報をレジスタ１３Ａに格納する。また、遅延量「３」に応じたＰＣ値、すなわち、そのときのＰＣの値に３×８を加算した値をレジスタ１３Ｂに格納する。ＤＢＲＡ命令の下に記述されている３命令は無条件に処理されるが、３番目の遅延スロットにある命令をフェッチするとき、マイクロプロセッサのＰＣの値はレジスタ１３Ｂに格納されているＰＣ値に一致する。そこで、ＰＣ制御部１３は、ＤＢＲＡ命令による分岐処理を実行する。すなわち、レジスタ１３Ｂに格納されているＰＣ値をマイクロプロセッサにおけるＰＣに設定する処理を行う。

【００５９】図１３はディレイド分岐命令を含むプログラムの他の例を示す説明図である。ここでは、サブルーチンコール命令であるＤＪＳＲ命令が用いられている。また、遅延量は、汎用レジスタ５におけるｒ４レジスタに設定されている値が用いられる。この場合も、ＰＣ制御部１３は、分岐先を示すＴＧＴ１に応じた情報をレジスタ１３Ａに格納する。また、ＰＣ制御部１３は、ｒ４レジスタに設定されている値に応じたＰＣ値をレジスタ１３Ｂに格納する。ｒ４レジスタに設定されている値が「４」であったとすると、ＤＪＳＲ命令の下に記述されている４命令は無条件に処理されるが、４番目の遅延スロットの命令をフェッチするときに、マイクロプロセッサのＰＣの値はレジスタ１３Ｂに格納されているＰＣ値に一致する。そこで、ＰＣ制御部１３は、ＤＪＳＲ命令によるサブルーチンジャンプを実行する。具体的には、ＰＣに、レジスタ１３Ｂに格納されているＰＣ値を設定する。

【００６０】以上のように、この実施の形態１によれば、マイクロプロセッサは、ディレイド分岐命令で指定された遅延量に応じたＰＣ値を保持するように構成され

ているので、ディレイド分岐命令のデコード時点から実行時点までの間に割り込み等のＰＣ値を変化させる事象が生じたとしても、確実に分岐命令が実行される。例えば、図１２に示された例において、遅延スロットの命令の実行時に割り込みが生ずるとマイクロプロセッサのＰＣの値は変化するとともに、割り込み処理において幾つかのサイクルが消費される。従来のマイクロプロセッサにおける処理のようにディレイド分岐命令の実行時期をカウンタ値として保持していると、割り込み処理において費やされたサイクル数分だけ実行時期がずれてしまう。しかし、この実施の形態によれば、そのようなことはない。また、レジスタ１３Ａ，１３Ｂをアクセスする命令を備えれば、遅延スロットの命令実行時に起きた割り込みにともないコンテクストスイッチが必要になった場合、レジスタ１３Ａに保持された分岐先情報とレジスタ１３Ｂに保持された分岐発生ＰＣ値とをデータＲＡＭ７に退避し、異なるコンテクストを実行後、再び割り込まれたコンテクストに復帰したときにこれらの値を復帰して遅延分岐を実行することも可能になる。

【００６１】実施の形態２．図１４はこの発明の実施の形態２によるマイクロプロセッサにおける命令デコード部分および命令実行部分を示すブロック図である。図において、３６１は算術論理演算を実行するＡＬＵ（演算ユニット）、３６３は乗算を実行する乗算器（演算ユニット）、３６５はＰＣ値を計算するＰＣ制御部、３６７はアドレス計算を行うメモリ制御部（演算ユニット）、３６９はシフト演算を実行するシフタ（演算ユニット）、３７１は１サイクルで２命令（２演算指令を含む命令）を転送できるバス、３７２は命令をデコードして命令実行部に制御信号１１，１２を与える命令デコーダ、３７３は汎用レジスタである。

【００６２】ＡＬＵ３６１において、３６２はディレイド算術論理演算命令のデコード結果を格納するレジスタ３６２Ａと遅延量に応じた値を格納するレジスタ３６２Ｂとを有する遅延命令レジスタ（遅延命令保持手段）である。乗算器３６３において、３６４はディレイド乗算命令のデコード結果を格納するレジスタ３６４Ａと遅延量に応じた値を格納するレジスタ３６４Ｂとを有する遅延命令レジスタ（遅延命令保持手段）である。ＰＣ制御部３６５において、３６６はディレイド分岐命令のデコード結果を格納するレジスタ３６６Ａと遅延量に応じた値を格納するレジスタ３６６Ｂとを有する遅延命令レジスタ（遅延分岐命令保持手段）である。メモリ制御部３６７において、３６８はディレイドメモリアクセス命令のデコード結果を格納するレジスタ３６８Ａと遅延量に応じた値を格納するレジスタ３６８Ｂとを有する遅延命令レジスタ（遅延命令保持手段）である。シフタ３６９において、３７０はディレイドシフト命令のデコード結果を格納するレジスタ３７０Ａと遅延量に応じた値を格納するレジスタ３７０Ｂとを有する遅延命令レジスタ（遅延命令保持手段）である。

【００６３】次に動作について説明する。図１５は各行の２演算を同時に実行する命令配置の一例を示す説明図である。この配置は、図２２に示されたプログラムをスケジューリングした結果である。図２１に示された一般的な２演算を同時に扱うマイクロプロセッサでは、図２２に示されたプログラムを実行するために、図２３に示されたように５サイクルを要していた。ＳＲＡ命令とＳＵＢ命令とはｒ３レジスタ依存の関係にあって同時に実行できないからである。ところが、この実施の形態によるマイクロプロセッサはディレイド分岐命令を扱えるので、図１５に示されたように、図２３における２行目のＮＯＰを遅延量「２」のＤＪＭＰ命令で置き換えることができる。

【００６４】図１５に示されたような命令配置による各命令を順次入力すると、第１サイクルでは、マイクロプロセッサにおいて、ＡＬＵ３６１とシフタ３６９とが、１行目のＡＤＤ演算とＳＲＡ演算とをそれぞれ実行する。第２サイクルでは、シフタ３６９とＰＣ制御部３６５とがＳＲＡ演算とＤＪＭＰ演算とを扱う。シフタ３６９はＳＲＡ演算を直ちに実行するが、ＰＣ制御部３６５は、実施の形態１の場合と同様に、ＤＪＭＰ演算のデコード結果である分岐先ＴＧＴに関する情報をレジスタ３６６Ａに格納し、遅延量「２」に応じたＰＣ値をレジスタ３６６Ｂに格納する。第３サイクルでは、ＡＬＵ３６１と乗算器３６３とが、３行目のＳＵＢ演算とＭＵＬ演算とをそれぞれ実行する。第４サイクルでは、ＡＬＵ３６１と乗算器３６３とが、４行目のＡＤＤ演算とＭＵＬ演算とをそれぞれ実行する。４行目の命令がフェッチされるとき、マイクロプロセッサのＰＣ（図１４において図示せず）の値はレジスタ３６６Ｂに格納されているＰＣ値に一致している。そこで、ＰＣ制御部３６５は、ＤＪＭＰ演算で指定されたジャンプ命令を実行する。具体的には、ＰＣに、レジスタ３６６Ｂに格納されているＰＣ値を設定する。

【００６５】このように、遅延量を任意に指定できる分岐命令処理する機構を備えていれば、命令のスケジューリングをより有効に実行することができる。すなわち、プログラマまたはコンパイラは、プログラムサイズをより小さくするような命令のスケジューリングを行える。その結果、ある処理を実行するのに要する時間は短縮される。しかも、ＰＣ制御部３６５は、分岐命令実行時期をＰＣ値として保持するので、ディレイド分岐命令は矛盾なく実行される。

【００６６】この実施の形態によるマイクロプロセッサは、ディレイド分岐命令以外のディレイド演算命令を扱うこともできる。図１６は、プログラムの一例を示す説明図である。図２１に示された一般的な２演算を同時に扱うマイクロプロセッサでは、図１６における１行目のＡＤＤ演算と２行目のＡＤＤ演算を同時実行することは

できない。1つのALU341しか持っていないからである。従って、図16における1行目の命令と2行目の命令とを同時実行するように命令のスケジューリングを行うことはできなかった。

【0067】ところが、この実施の形態によるマイクロプロセッサは、図16に示すプログラムの各命令を図17に示すようにスケジューリングした各命令を実行できる。図17に示されたような命令配置による各命令を順次入力すると、このマイクロプロセッサにおいて、第1サイクルでは、ALU361は、ADD演算を実行するとともにDADD（ディレイドADD）演算も扱う。すなわち、DADDのデコード結果をレジスタ362Aに格納するとともに、遅延値「1」に応じた値をレジスタ362Bに格納する。第2サイクルでは、シフタ369と乗算器363とは、SRA演算とMUL演算とをそれぞれ実行する。2行目の命令がフェッチされるときに、PCの値はレジスタ362B内の遅延値「1」に応じた値に一致する。そこで、ALU361は、レジスタ362AからDADDで指定された演算指令内容を取り出し、その演算を実行する。

【0068】なお、レジスタ362Bに格納される遅延値に応じた値は、動作クロックに応じて値が変化するカウンタ値であってもよいが、PC値であってもよい。遅延値に応じた値としてPC値を保持した場合には、デコード時と実行時との間で割り込み等が生じてもそれに対する対処が容易になる。

【0069】ここでは、ディレイド演算命令を扱う演算ユニットとしてALU361を例にとったが、乗算器363、メモリ制御部367およびシフタ369も、ALU361の処理と同様の処理によってディレイド演算命令を扱うことができる。また、遅延値「1」の場合を例に説明したが、もちろん、任意の値を指定できる。任意の値を指定可能なので、命令のスケジューリングの自由度をより向上させることができる。なお、固定的な値を指定するディレイド演算命令しか処理しないマイクロプロセッサであっても、任意の遅延値を指定できるディレイド演算命令を処理するマイクロプロセッサに比べると効果は低減するものの、命令のスケジューリングの自由度を向上させる効果が期待できる。

【0070】以上のように、ディレイド演算命令を扱う機構を有する場合には、プログラムの実行に要するサイクル数をさらに少なくでき、その結果、プログラム実行時間を短縮できる。例えば、図16に示されたプログラムがあった場合に、図21に示された一般的な2演算を同時に扱うマイクロプロセッサでは、2つのADD命令を同時実行できないので、命令のスケジューリングを行ってもプログラム実行に3サイクルを要する。このマイクロプロセッサは、図17に示された例からわかるように、2サイクルでプログラムを実行できる。また、ループ内でディレイド命令を記述できるような場合に、このマイクロプロセッサによれば、さらに処理を高速化できる。

【0071】ディレイド命令は、2演算同時実行できるマイクロプロセッサにおいて、特に有効である。図15および図17に示された例からわかるように、演算を並列実行する場合のNOP挿入箇所を削減できるからである。なお、ここでは、2演算同時実行のマイクロプロセッサについて説明したが、リソースであるALU361等の演算ユニットを複数個備え、同時実行演算数をさらに多くしたマイクロプロセッサに、この実施の形態によるディレイド演算命令処理機構を設けてもよい。そのようなマイクロプロセッサにおいても、実行サイクル数の削減および処理の高速化が期待できる。

【0072】図18はこの実施の形態2によるマイクロプロセッサにおけるPSWの一例を示す説明図である。このPSW380，390は図5に示されたものと同様のものであるが、この場合には、RPフィールド381がPC制御部365の遅延命令レジスタ366の動作を有効にするかどうか決めるビットとして用いられる。また、例えば、E4フィールド382、E3フィールド383、E2フィールド384およびE1フィールド385は、それぞれ、ALU361の遅延命令レジスタ362、乗算器363の遅延命令レジスタ364、メモリ制御部367の遅延命令レジスタ368およびシフタ369の遅延命令レジスタ370の動作を有効にするかどうか決めるビットとして用いられる。各ビットが無効状態にセットされた場合には、対応する演算ユニットにおけるディレイド命令処理は禁止される。このようなPSW380，390を用意することによって、ディレイド命令処理の柔軟性を増すことができる。

【0073】実施の形態3．図19はこの発明の実施の形態3によるマイクロプロセッサにおける命令デコード部分および命令実行部分を示すブロック図である。図において、461は算術論理演算を実行するALU（演算ユニット）、463は乗算を実行する乗算器（演算ユニット）、465はPC値を計算するPC制御部、467はアドレス計算を行うメモリ制御部（演算ユニット）、469はシフト演算を実行するシフタ（演算ユニット）、371は1サイクルで2命令を転送できるバス、372は命令をデコードして命令実行部に制御信号11，12を与える命令デコーダ、373は汎用レジスタである。

【0074】このマイクロプロセッサにおいて、ALU461は、2つの遅延命令レジスタ362を有する。乗算器463は、2つの遅延命令レジスタ364を有する。PC制御部465は、2つの遅延命令レジスタ366を有する。メモリ制御部467は、2つの遅延命令レジスタ368を有する。そして、シフタ469は、2つの遅延命令レジスタ370を有する。

【0075】次に動作について説明する。各演算ユニッ

トおよびPC制御部465において、命令のデコード結果と遅延量とからなる1セットはキュー管理される。例えば、ALU461において、一方の遅延命令レジスタ362のレジスタ362Aにディレイド算術演算命令のデコード結果が格納されレジスタ362Bにそれに対応する遅延量に応じた値が設定されている場合に、さらに、命令デコーダ372から新たなディレイド算術演算命令のデコード結果が送られてきたとする。すると、新たなデコード結果および遅延量に応じた値は、他方の遅延命令レジスタ362に格納される。そして、一方のレジスタ362Bに格納されている遅延量に応じた値とマイクロプロセッサのPCの値とが一致したときに、一方のレジスタ362Aに格納されているデコード結果に応じた演算が実行される。その後、他方のレジスタ362Bに格納されている遅延量に応じた値とマイクロプロセッサのPCの値とが一致したときに、他方のレジスタ362Aに格納されているデコード結果に応じた演算が実行される。

【0076】そのような構成によれば、あるディレイド命令の実行時期に到達していない時点でも、別の同種類のディレイド命令を扱うことができる。例えば、あるディレイドADD命令の実行時期に到達していない時点でも、ALU461は、新たなディレイドADD命令を受け入れることができる。従って、命令のスケジューリングの自由度をさらに上げることができ、プログラムサイズの減少すなわち命令メモリの容量削減を図ることができる。また、より高速にプログラムを実行することができる。なお、ここでは各演算ユニットにおいて2つの遅延命令レジスタが設けられている例を示したが、設置数をさらに多くしてもよい。

【0077】

【発明の効果】以上のように、請求項1記載の発明によれば、遅延命令を有するマイクロプロセッサを、ディレイド分岐命令で指定された遅延量に応じたプログラムカウンタ値を保持するように構成したので、ディレイド分岐命令のデコード時点から実行時点までの間に割り込み等のPCの値を変化させる事象が生じたとしても確実に分岐命令が実行される効果がある。

【0078】請求項2記載の発明によれば、遅延命令を有するマイクロプロセッサを、各演算ユニットがディレイド演算命令の固定のまたは可変の遅延量に応じた値を保持するように構成したので、命令スケジューリングの自由度を向上させることができる効果がある。

【0079】請求項3記載の発明によれば、遅延命令を有するマイクロプロセッサを、各演算ユニットがディレイド演算命令で指定された任意の遅延量に応じた値を保持するように構成したので、プログラムの実行に要するサイクル数をより少なくでき、その結果、命令メモリの容量を削減できるとともにプログラム実行時間を短縮できる効果がある。

【0080】請求項4記載の発明によれば、遅延命令を有するマイクロプロセッサを、各演算ユニットがディレイド演算命令で指定された遅延量に応じたPC値を保持するように構成したので、プログラムの実行に要するサイクル数をより少なくできるとともに、ディレイド演算命令のデコード時点から実行時点までの間に割り込み等のPCの値を変化させる事象が生じたとしても確実に演算が実行される効果がある。

【0081】請求項5記載の発明によれば、遅延命令を有するマイクロプロセッサを、各演算ユニットに複数の遅延命令保持手段が設けられるように構成したので、命令のスケジューリングの自由度を上げることができる効果がある。

【0082】請求項6記載の発明によれば、遅延命令を有するマイクロプロセッサを、命令実行部が複数演算を同時に実行するように構成したので、命令のスケジューリングの自由度をさらに上げることができ、その結果、プログラムの実行に要するサイクル数をさらに少なくできる効果がある。

【図面の簡単な説明】

【図1】　この発明の実施の形態1によるマイクロプロセッサの構成を示すブロック図である。

【図2】　マイクロプロセッサの命令フォーマットを示す説明図である。

【図3】　演算フィールドの詳細な内容を示す説明図である。

【図4】　マイクロプロセッサのレジスタ構成を示す説明図である。

【図5】　PSWの詳細内容を示す説明図である。

【図6】　マイクロプロセッサの並列2命令実行時のパイプライン動作を示す説明図である。

【図7】　マイクロプロセッサのシーケンシャル命令実行時のパイプライン動作を示す説明図である。

【図8】　この発明の実施の形態1によるマイクロプロセッサの構成を示すブロック図である。

【図9】　ディレイド分岐命令の基本的なフォーマットを示す説明図である。

【図10】　幾つかのディレイド分岐命令の例を示す説明図である。

【図11】　ディレイド分岐命令、ディレイドジャンプ命令およびディレイドサブルーチンコール命令が、同一フォーマットでどのように実現されるのかを示すための説明図である。

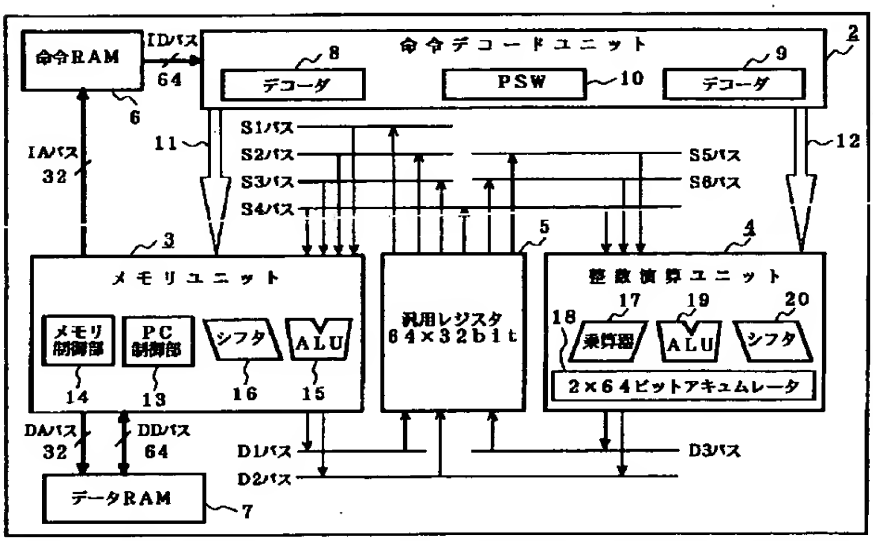
【図12】　ディレイド分岐命令を含むプログラムの一例を示す説明図である。

【図13】　ディレイド分岐命令を含むプログラムの他の例を示す説明図である。

【図14】　この発明の実施の形態2によるマイクロプロセッサにおける命令デコード部分および命令実行部分を示すブロック図である。

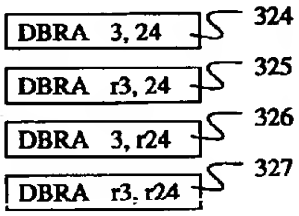
【図15】　2演算を同時に実行する命令配置の一例を示す説明図である。
【図16】　プログラムの一例を示す説明図である。
【図17】　図16に示された各命令にもとづく2演算を同時に実行する命令配置を示す説明図である。
【図18】　この発明の実施の形態2によるマイクロプロセッサにおけるPSWの一例を示す説明図である。
【図19】　この発明の実施の形態3によるマイクロプロセッサにおける命令デコード部分および命令実行部分を示すブロック図である。
【図20】　従来のパイプライン制御方式によるマイクロプロセッサの処理シーケンスを示すシーケンス図である。
【図21】　2演算を同時に行うマイクロプロセッサにおける命令デコーダおよび命令実行部の部分の一般的な構成を示すブロック図である。
【図22】　プログラムの一例を示す説明図である。
【図23】　命令のスケジューリングの一例を示す説明図である。
【符号の説明】
2　命令デコードユニット（命令デコーダ）、3　メモリユニット（命令実行部）、4　整数演算ユニット（命令実行部）、13，365，465　PC制御部、361，461　ALU（演算ユニット）、362，364，368，370　遅延命令レジスタ（遅延命令保持手段）、363，463　乗算器（演算ユニット）、366　遅延分岐命令レジスタ（遅延分岐命令保持手段）、367，467　メモリ制御部（演算ユニット）、369，469　シフタ（演算ユニット）。

【図1】



2:命令デコードユニット(命令デコーダ)　3:メモリユニット(命令実行部)　4:整数演算ユニット(命令実行部)

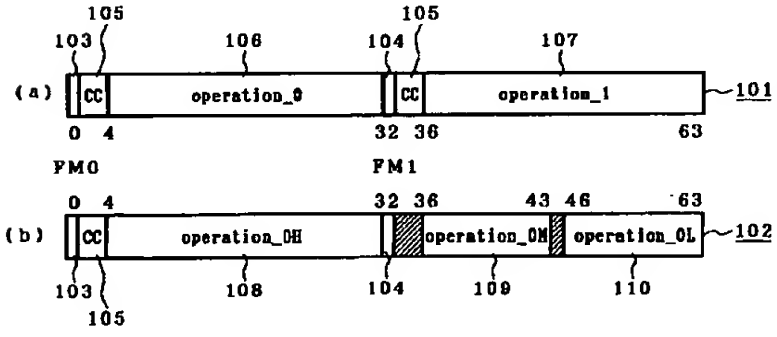
【図10】



【図16】

```
ADD  r3, r10, 6
ADD  r4, r4, 8
SRA  r8, r3, 1
MUL  r5, r3, 4
```

【図2】



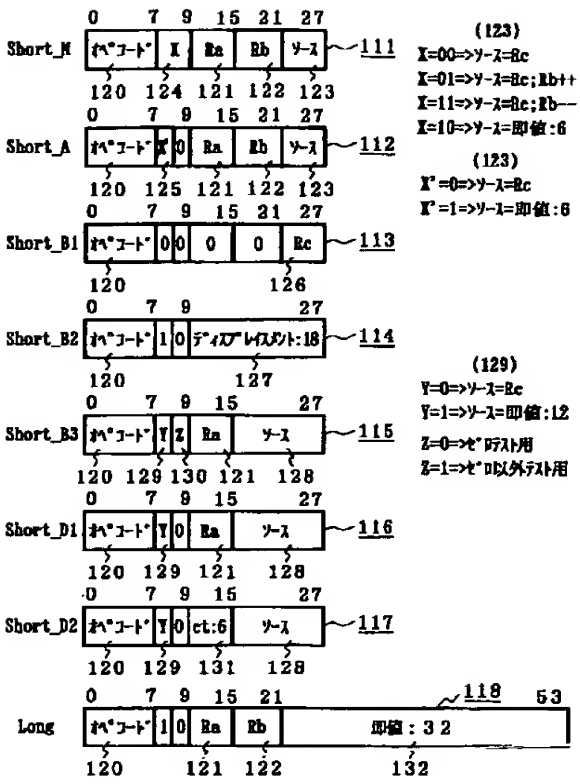
【図15】

```
ADD  r3, r0, 6      SRA  r4, r0, 8
SRA  r3, r3, 1      DJMP 2, TGT
SUB  r4, r4, r3     MUL  r5, r3, 4
ADD  r4, r4, r10    MUL  r11, r11, r5
```

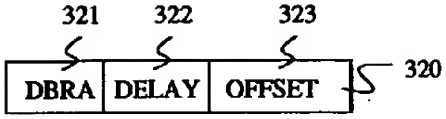
【図17】

```
ADD  r3, r10, 6   DADD  1, r4, r4, 8
SRA  r8, r3, 1    MUL   r5, r3, 4
```

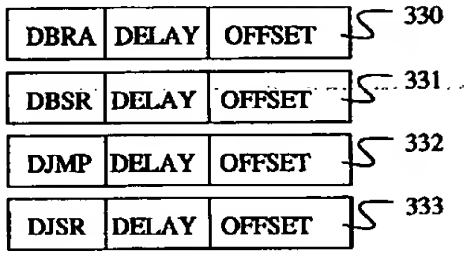
【図3】



【図9】



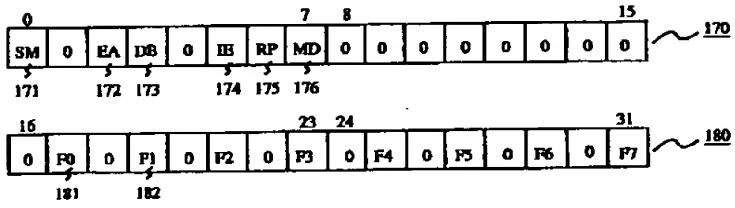
【図11】



【図22】

```
ADD  r3, r0, 6
SRA  r4, r0, 8
SRA  r3, r3, 1
SUB  r4, r4, r3
MUL  r5, r3, 4
ADD  r4, r4, r10
MUL  r11, r11, r5
JMP  TGT
```

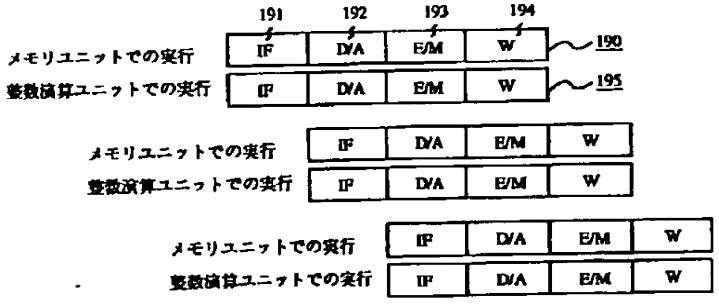
【図5】



【図23】

```
ADD  r3, r0, 6      SRA  r4, r0, 8
SRA  r3, r3, 1      NOP
SUB  r4, r4, r3     MUL  r5, r3, 4
ADD  r4, r4, r10    MUL  r11, r11, r5
JMP  TGT
```

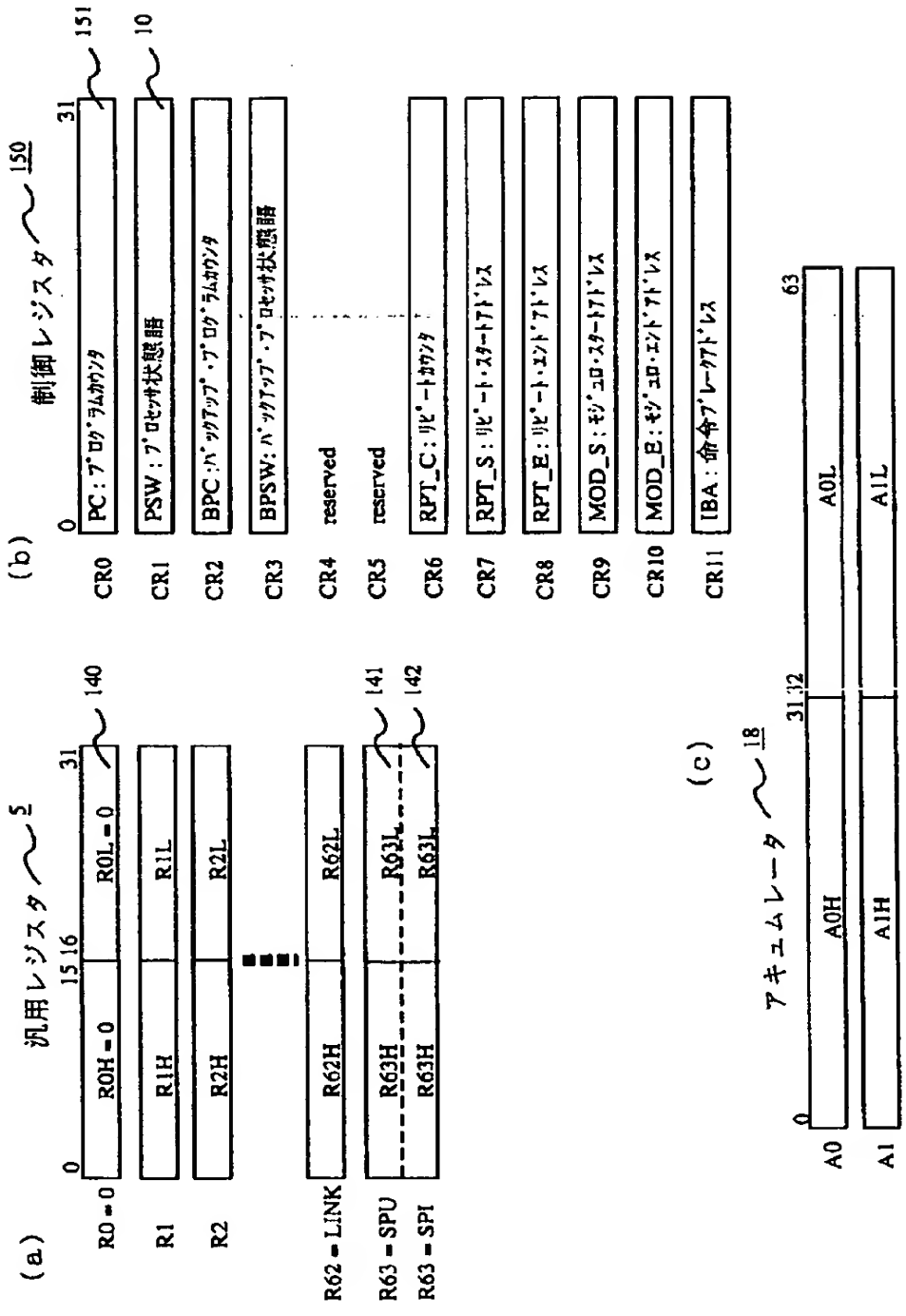
【図6】



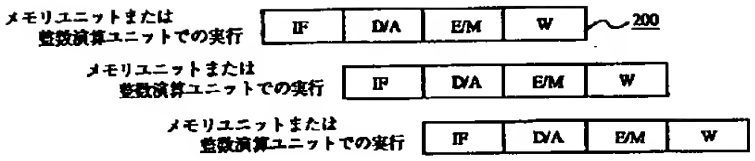
【図12】

```
      ADD   r3, r4, r5
      DBRA  3, TGT1
      SUB   r3, r4, 10  ←— delay slot 1
      SRA   r1, r2, r3  ←— delay slot 2
      ADD   r3, r4, 11  ←— delay slot 3
      ...
TGT1: ADD   r3, r4, 20
      ...
```

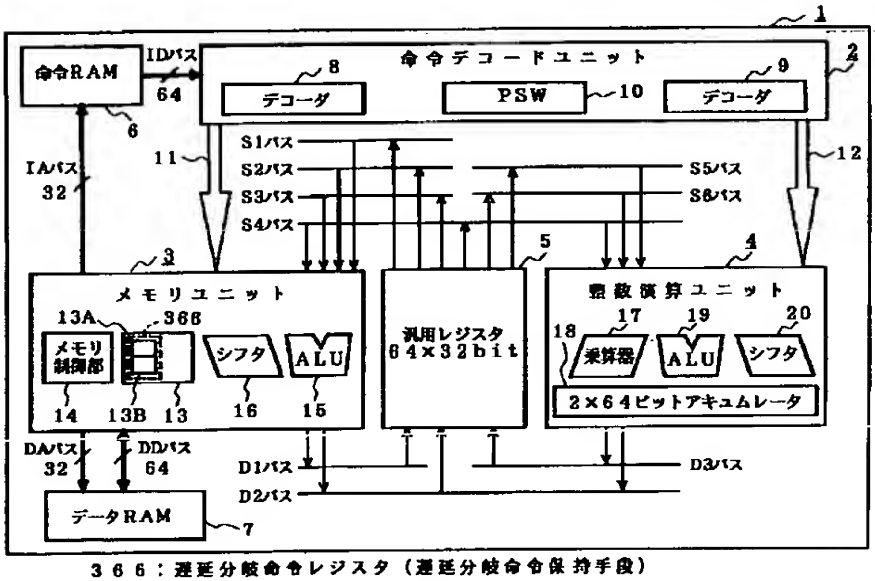
【図4】


(a)
汎用レジスタ 5
140
31
15 16
0
R0H = 0
R0L = 0
R1H
R1L
R2H
R2L
R62H
R62L
R63H
R63L
141
142
R0 = 0
R1
R2
R62 = LINK
R63 = SPU
R63 = SPI
(b)
制御レジスタ 150
151
10
31
0
PC : ﾌﾟﾛｸﾞﾗﾑｶｳﾝﾀ
PSW : ﾌﾟﾛｾｯｻ状態語
BPC : ﾊﾞｯｸｱｯﾌﾟ・ﾌﾟﾛｸﾞﾗﾑｶｳﾝﾀ
BPSW : ﾊﾞｯｸｱｯﾌﾟ・ﾌﾟﾛｾｯｻ状態語
reserved
reserved
RPT_C : ﾘﾋﾟｰﾄｶｳﾝﾀ
RPT_S : ﾘﾋﾟｰﾄ・ｽﾀｰﾄｱﾄﾞﾚｽ
RPT_E : ﾘﾋﾟｰﾄ・ｴﾝﾄﾞｱﾄﾞﾚｽ
MOD_S : ﾓｼﾞｭﾛ・ｽﾀｰﾄｱﾄﾞﾚｽ
MOD_E : ﾓｼﾞｭﾛ・ｴﾝﾄﾞｱﾄﾞﾚｽ
IBA : 命令ﾌﾞﾚｰｸｱﾄﾞﾚｽ
CR0
CR1
CR2
CR3
CR4
CR5
CR6
CR7
CR8
CR9
CR10
CR11
(c)
アキュムレータ 18
63
31 32
0
A0H
A0L
A1H
A1L
A0
A1

【図7】



【図8】

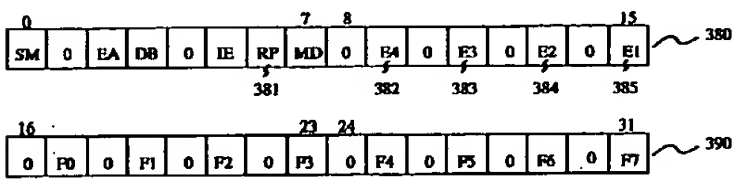


366：遅延分岐命令レジスタ（遅延分岐命令保持手段）

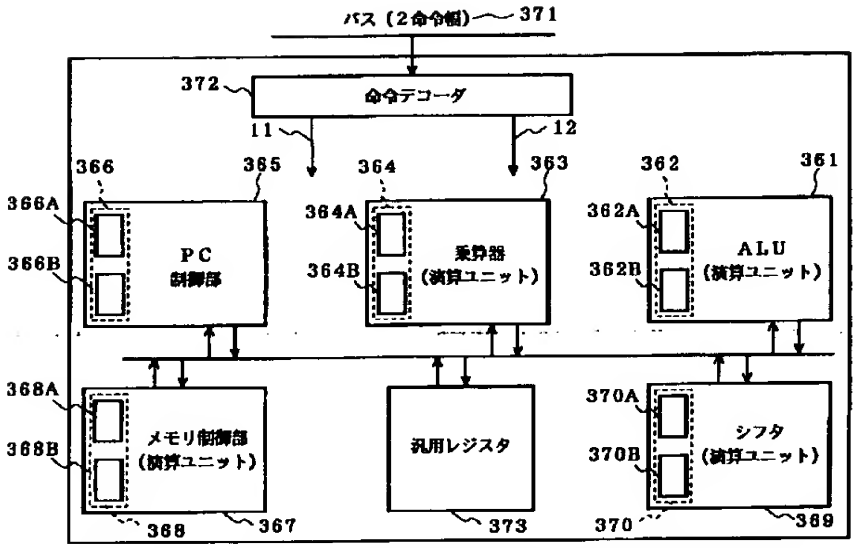
【図13】

```
        ADD   r4, r0, 4
        ADD   r3, r0, 5
        DJSR  r4, TGT1
        SUB   r3, r4, 10  ←― delay slot 1
        SRA   r1, r2, r3  ←― delay slot 2
        ADD   r3, r4, 11  ←― delay slot 3
        ADD   r8, r8, 2   ←― delay slot 4
        ...
TGT1:   ADD   r3, r4, 20
        ...
        RSR
```

【図18】

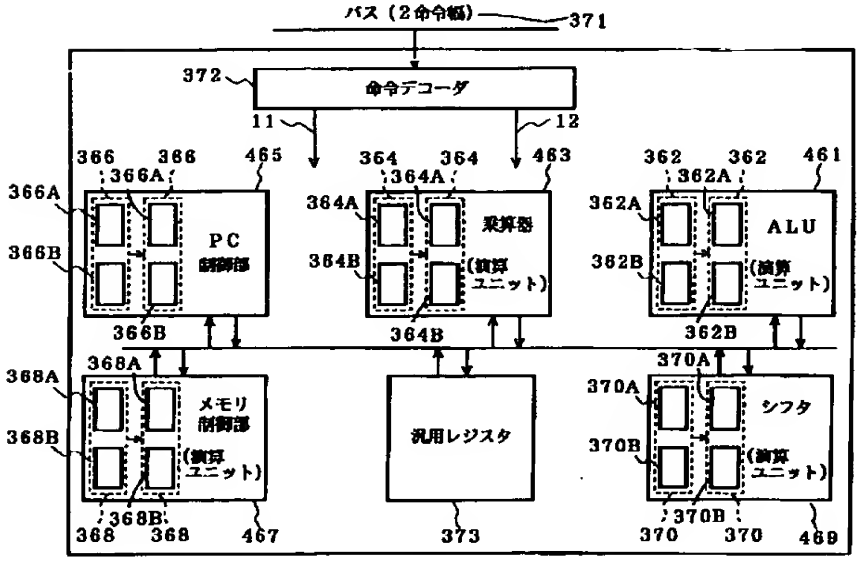


【図14】

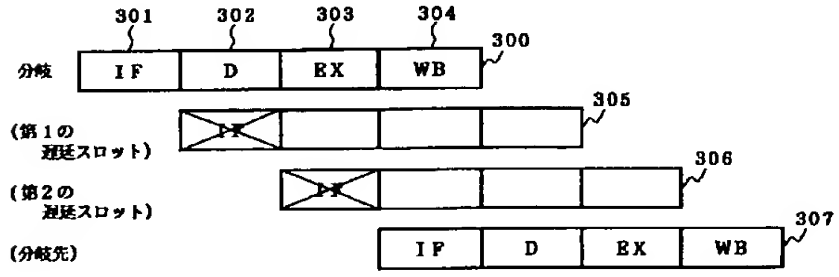


362，364，368，370：遅延命令レジスタ（遅延命令保持手段）

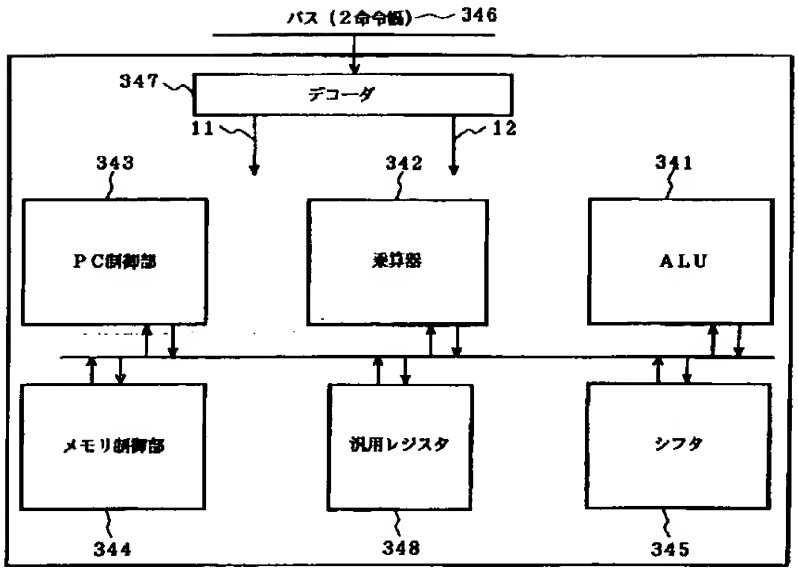
【図19】



【図20】



【図21】


バス（2命令幅）
346
347
デコーダ
11
12
343
342
341
PC制御部
乗算器
ALU
メモリ制御部
汎用レジスタ
シフタ
344
348
345